

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 07-281897

(43)Date of publication of application : 27.10.1995

(51)Int.Cl.

G06F 9/42

G06F 9/46

(21)Application number : 07-108056

(71)Applicant : HEWLETT PACKARD CO <HP>
HITACHI LTD

(22)Date of filing : 06.04.1995

(72)Inventor : AMERSON FREDERIC C
ENGLISH ROBERT M
GUPTA RAJIV
WATANABE HIROSHI

(30)Priority

Priority number : 94 223804 Priority date : 06.04.1994 Priority country : US

(54) REGISTER ASSIGNMENT METHOD AND REGISTER FILE PORT ACCESS DEVICE

(57)Abstract:

PURPOSE: To reduce the amount of save and restore of a register at the time of a procedure call/return.

CONSTITUTION: Only registers of a truly required number of registers assigned from a register pool for each procedure by improving the register window system. This assignment is made by a called procedure itself independently of a caller procedure. When the registers in the register pool are used out, part of contents of the registers assigned already is saved in a memory to make an idle register.

LEGAL STATUS

[Date of request for examination] 19.03.2002

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number] 3529888

[Date of registration] 05.03.2004

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

Copyright (C); 1998,2003 Japan Patent Office

BEST AVAILABLE COPY

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平7-281897

(43) 公開日 平成7年(1995)10月27日

(51) IntCl. ⁸	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F 9/42	3 3 0 R			
9/46	3 1 3 B	7737-5B		

審査請求 未請求 請求項の数10 F D (全 15 頁)

(21) 出願番号	特願平7-108056	(71) 出願人	590000400 ヒューレット・パッカード・カンパニー アメリカ合衆国カリフォルニア州パロアル ト ハノーバー・ストリート 3000
(22) 出願日	平成7年(1995)4月6日	(71) 出願人	000005108 株式会社日立製作所 東京都千代田区神田駿河台四丁目6番地
(31) 優先権主張番号	2 2 3, 8 0 4	(72) 発明者	フレドリック・シー・アマーソン アメリカ合衆国カリフォルニア州サンタ・ クララ、ヒルズデイル・アベニュー 451 アパートメント・ナンバー・ビー
(32) 優先日	1994年4月6日	(74) 代理人	弁理士 上野 英夫
(33) 優先権主張国	米国 (U S)		

最終頁に続く

(54) 【発明の名称】 レジスタ割当て方法及びレジスタファイルポートアクセス装置

(57) 【要約】

【目的】 プロシージャコール、リターンの際のレジスタのセーブ、リストアの量を減らす。

【構成】 レジスタウインドウ方式を改良して、レジスタプールからプロシージャ毎に真に必要な個数のレジスタだけを割り当てる。この割り当ては、呼び出されたプロシージャが呼出し元のプロシージャとは独立に自分で割り当てを行う。レジスタプール中のレジスタを使い切ったら、既に割り当てられているレジスタの内容の一部をメモリにセーブして、空きを作る。

【特許請求の範囲】

【請求項1】物理レジスタセットを有するデジタルコンピュータにおいて、コンパイラの介在なしに動的にレジスタをプロシージャに割り当てる方法であって、複数のスタックレジスタを有する論理レジスタスタック(58)を定義するステップと、

前記論理レジスタスタックを前記物理レジスタセット

(図4A)にマッピングするためのオフセットを定義するためにローカルリロケーション項(lrel)を初期化するステップと、

第1のスタックポインタ値(TOL)を初期化することにより、第1のプロシージャにより指定された任意の個数のスタックレジスタ(62)をローカルレジスタとして第1のプロシージャ(A)に割り当てて、前記論理レジスタスタック内の前記ローカルレジスタ(62)を区切るステップと、

前記第1のプロシージャ実行の間のレジスタアクセス操作に関連して、前記論理リロケーション項(lrel)に応答して各ローカルレジスタ論理アドレス(R、図5)を前記物理レジスタセット(r)にマッピングするステップを設けたことを特徴とするレジスタ割当て方法。

【請求項2】前記第1のスタックポインタ値(VOL)をストアして第2のスタックポインタ値(OTOL)を形成するステップと、

前記第1のスタックポインタ値(TOL)をインクリメントすることにより、前記第1のプロシージャにより指定された任意の個数の追加のスタックレジスタ(64)をパラメータ引渡レジスタとして前記第1のプロシージャ(A)に割り当て、パラメータ引渡レジスタを含ませるようにするステップと、

呼び出されたプロシージャ(B)が参照するために、選択されたパラメータを前記割り当てられたパラメータ引渡レジスタ(64)にストアするステップであって、前記ストアするステップがパラメータ引渡レジスタを前記ローカルリロケーション項(図5)に응答して前記物理レジスタセットにマッピングするステップを含むようなステップを設けたことを特徴とする請求項1記載のレジスタ割当て方法。

【請求項3】第2のプロシージャ(B)を呼び出し、第1のプロシージャパラメータ引渡レジスタ(64)を有する初期ローカルレジスタ空間を前記第2のプロシージャに割り当て、これにより前記レジスタにストアされた前記選択されたパラメータを、メモリ参照なしに前記第2のプロシージャに利用可能にするステップと、

前記スタックポインタ値(TOL)をインクリメントすることにより、前記第2のプロシージャにより指定される任意の個数の追加のスタックレジスタをローカルレジスタ(66)として前記第2のプロシージャ(A)に割り当てて、前記第1プロシージャのローカルレジスタの

内容をはじめにメモリにセーブすることなしに、前記第2のプロシージャのローカルレジスタ(66)を含ませるようにするステップと、前記第2のプロシージャから復帰したとき、前記スタックポインタ値(TOL)をローカルレジスタ(66)の個数だけデクリメントすることにより前記ローカルレジスタを割り当て解除して、ローカルレジスタ内容をセーブ及び復元することなく前記第2のプロシージャを呼び出したそこから復帰するステップを設けたことを特徴とする請求項2記載のレジスタ割当て方法。

10

【請求項4】前記第2のプロシージャを呼び出したとき前記第1及び第2のスタックポインタ値(TOL, OTOL)をストアして、前記第2のプロシージャから復帰した際に参照するためのストアされた値(制御レジスタA)を形成するステップを設け、

ここにおいて、前記割り当て解除するステップが前記第1及び第2のスタックポインタ値を前記ストアされた値にリセットするステップを含むことを特徴とする請求項3記載のレジスタ割当て方法。

20

【請求項5】追加のパラメータレジスタを前記第1のプロシージャに割り当てるステップを設け、

ここにおいて、前記ストアするステップが、前記第2のプロシージャからの復帰の際に参照するために前記追加のパラメータレジスタにスタックポインタオフセット値をストアするステップを含むことを特徴とする請求項2, 3または4記載のレジスタ割当て方法。

【請求項6】ローカル下限(BOL)ポインタ値を初期化して、現プロシージャに割り当てられた一連のスタックレジスタの一方の端を指示し、前記現プロシージャに割り当てられた一連のスタックレジスタの他方の端が前記第1のスタックポインタ値(TOL)により指示され、

30

ここにおいて、前記ローカルリロケーション項(lrel)が、前記ローカル下限ポインタ値から予め定められた定数個のスタックレジスタを引いたものに等しいことを特徴とする請求項2, 3または4記載のレジスタ割当て方法。

【請求項7】前記レジスタスタックを介してアクセス可能なソフトウェアスタックの深さを指示するために有効範囲下限ポインタ(BOV)を初期化するステップを含み、

40

ここで前記第1のスタックポインタ値(TOL)をインクリメントすることがモジュール加算及びモジュール物理レジスタ数を使用して行われ、それにより前記レジスタセットがリングとして管理されるようにし、更に、前記第1のスタックポインタのインクリメントが有効範囲下限ポインタ(BOV)値より大きい値を招くとき、レジスタオーバーフロー状態を指示するステップを含むことを特徴とする請求項2, 3または4に記載の方法。

50

【請求項8】第1の循環レジスタポインタ値(BOR)

及び第2の循環レジスタポインタ値(TOR)を前記第1のスタックポインタ値(TOL)に初期化するステップと、

前記第2の循環レジスタポインタ値(TOR)及び前記第1のスタックポインタ値(TOL)を、前記呼び出されたプロシーダにより循環レジスタとして指定されたレジスタの任意の個数だけインクリメントすることにより、レジスタを呼び出されたプロシーダ(C, 図3F)に循環レジスタ(74)として割り当てるステップと、

前記呼び出されたプロシーダから復帰するに先立って、前記第2の循環レジスタポインタ値及び前記第1スタックポインタ値を循環レジスタの個数だけデクリメントすることにより、前記循環レジスタの割当てを解除するステップを設けたことを特徴とする請求項1記載のレジスタ割当て方法。

【請求項9】請求項1に記載の方法を実現するためのレジスタファイルポートをアクセスする物理アドレスを提供するためのレジスタファイルポートアクセス装置(140)において、

現プロシーダから仮想アドレス(R)を受け取る入力手段(142)と、

前記仮想アドレス(R)を予め設定された定数と比較して、前記仮想アドレスがスタティックレジスタを指示するのかそれともスタックレジスタを指示するのかを判定する比較手段(150)と、

前記仮想アドレスをローカルリロケーション項(1rel)に加算して、第1の物理アドレスを形成する手段(154)と、

前記仮想アドレス(R)と前記第1の物理アドレスの何れか一方を選択し、選択されたアドレス(r)を前記レジスタファイルポートに結合するマルチプレクサ手段(146)と、

前記マルチプレクサ手段に結合され、前記仮想アドレス(R)がスタックレジスタを指示している場合には前記第1の物理アドレスを選択し、前記仮想アドレスがスタティックレジスタを指示している場合には前記仮想アドレスを選択し、これによりスタックレジスタへの参照を前記現プロシーダに割り当てられた物理レジスタアドレスに振り替える前制御手段(152)を設けたことを特徴とするレジスタファイルポートアクセス装置。

【請求項10】前記仮想アドレス(R)と第1及び第2の循環レジスタポインタ値(BOR, TOR)とを比較して、前記仮想アドレスが前記現プロシーダに循環レジスタとして割り当てられたレジスタを指示するかどうかを判定する比較手段(164, 166)と、

第1の物理レジスタを形成するために前記仮想アドレス(R)を第1の循環リロケーション項(rrel)に加算する手段(170)と、

前記仮想アドレス(R)を第2の循環リロケーション項

(rrel#)に加算して、第2の物理レジスタを形成する手段(172)と、

前記第1及び第2の物理アドレスの何れか一方を選択し、選択されたアドレス(r)を前記レジスタファイルポートアドレス端子(144)に結合するマルチプレクサ手段(162)と、

前記マルチプレクサ手段(162)を制御して、前記仮想アドレス(R)が前記循環レジスタセット内での回り込み(wraparound)を起こさない場合には前記第1の物理アドレスを選択させ、前記仮想アドレスが前記循環レジスタセット内で回り込みを起こす場合には前記第2の物理アドレスを選択させる制御手段(168, 152)を設け、

前記第1の循環リロケーション項(rrel)は前記ローカルリロケーション項(1rel)に循環レジスタベース値(RRB)を加算した結果に等しく、前記第2の循環リロケーション項(rrel#)は前記ローカルリロケーション項(1rel)に前記循環レジスタベース値(RRB)から前記循環レジスタセットのサイズを差し引いたものを加算した結果に等しく、これにより前記循環レジスタセット内での前記回り込みを調整することを特徴とする請求項9記載のレジスタファイルポートアクセス装置。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は一般にデジタルコンピュータに関し、特にデジタルコンピュータのレジスタの管理に関する。

【0002】

【従来技術及びその問題点】ノイマン型デジタルコンピュータは操作中に種々の値を保持するためのレジスタセットを備えている。レジスタセットの大きさは可変である。全てのノイマン型装置は少なくともプログラムカウンタ(PC)を備えている。一般に、オペランド及び結果を保持するためのレジスタ(“演算レジスタ”)も備えている。RISC(縮小命令セットコンピュータ)マシンは、一般に、メモリへの書込み・メモリからの読出しを行うがデータに対する操作は行わないロード(LOAD)及びストア(STORE)命令以外では、レジスタからレジスタへの命令(メモリに直接アクセスする命令から区別される)しか持っていない。RISCマシンは大きなレジスタセットを持つ傾向があり、例えば32個またはもっと多くにも達する。レジスタは、中間結果、アドレスインデックス及びサブルーチンのような呼び出し元と呼び出し先のプロシーダ間の引渡データ

(パラメータ)を保持するために使用される。幾つかのプロセッサは汎用レジスタに加えて浮動小数点レジスタを備えている。CISCアーキテクチャは、通常、評価スタックを備えており、評価スタックはオペランドが明示的には示されない0アドレス演算を提供する。RISC

Cは、通常、評価スタックを備えていない。コンパイラは、RISCアーキテクチャにおいては、演算のためではなく、通常はパラメータ引渡し及びレジスタ内容の追出し(register spills)のために、スタックをメモリ内に保持する。殆どのアーキテクチャにおいて、プロシージャ呼び出しに当たってのレジスタのセーブと復元のオーバーヘッドは負担になる。このオーバーヘッドはメインメモリ参照の5%から40%を占める。このオーバーヘッドを減らすために、いくつかのバンクを設け、新たなバンクを各呼び出されたプロシージャに割り当てること
10 が知られている。この技術はレジスタウインドウと称される。J. Hennessy, D. Patterson著のComputer Architecture — a Quantitative Approach(1990) Section 8.7、を参照のこと。レジスタウインドウを使用することにより、これらのレジスタバンクつまり“ウインドウ”を、パラメータを引き渡すための共通エリアを提供するためにオーバーラップさせる。レジスタはプロシージャ
20 呼び出しに当たって変化しないグローバル(global)レジスタと、プロシージャ呼び出しに当たって変化するローカルレジスタに分けられる。バッファが一杯になった後に、プロシージャ呼び出しが行われたとき(ウインドウオーバーフロー)、またはバッファが空になった後にプロシージャからの復帰が起こったとき(ウインドウアンダーフロー)、レジスタのブロックはメモリにセーブされる。

【0003】レジスタウインドウは、現在、サンマイクロシステムズのSPARC(登録商標)アーキテクチャで実現されており、更にレジスタウインドウをリング構成で操作することを開示している米国特許第5,159,680号で説明されている。米国特許第5,233,691号は、バス競合
30 を起こさない時間にメモリにレジスタを予め書き込むことにより、オーバーフローによる書き込みを行う必要性を減らすレジスタウインドウシステムを開示している。オーバーラッピングウインドウを実現する高性能レジスタファイルは米国特許第5,226,142号及び米国特許第5,226,128号に開示されており、米国特許第5,083,267号及び米国特許第5,036,454号はループのために循環レジスタを使用することを開示している。

【0004】レジスタウインドウのような従来技術のアーキテクチャが有する1つの問題は、レジスタのバンク(すなわちレジスタウインドウ)のサイズが固定されていることである。このサイズはプロシージャ毎に変化させることができない。その結果、プロシージャに割り当てられたローカルレジスタエリア内の全てのレジスタが
40 実際にそのプロシージャにより使用されるわけではなく、逆に、多くの場合、プロシージャにそのプロシージャが必要とする十分なレジスタが割り当てられているわけではない。これにより、メモリ参照が最適化されないために、パフォーマンスの低下が引き起こされる。

【0005】レジスタウインドウの他の制限は、オーバーラップレジスタの個数もまた固定されていることであ
50

る。この場合にも、その個数は実際に呼び出されるプロシージャが必要とするパラメータの数を十分に上回ることがあり、一方ではレジスタの使用密度が減る。その上、この固定されたオーバーラップにより、単一のプロシージャ呼び出しに関して引渡されるパラメータの個数が勝手に限定される。

【0006】循環レジスタ(rotating register)空間は、あるデータを使用する操作が呼び出される何サイクルか前にそのデータの準備を開始しそのデータが必要になったときにはそれが使用可能になっているようにするために、ソフトウェアのパイプライン化されたループによって使用される。ソフトウェアのパイプライン化されたループ内で必要とされるレジスタの個数は、ループの特性により変わる。もし、従来技術のように循環レジスタ空間のサイズが固定されているなら、大多数のループを扱うために例えば64個のレジスタのような広大な空間を割り当てなければならない。しかし、16個またはそれ以下のレジスタしか必要としない多くの小ループがあり、また64個よりも多くのレジスタを必要とする多くの大ループがある。小ループの場合には、多くのレジスタが不必要に割り当てられまた開放され、大ループの場合には、レジスタの不足により処理速度が低下する。

【0007】上記事情に鑑み、従来技術のレジスタウインドウのグループサイズが固定されているという枠組みに制約されない、レジスタの効率的な割り当て及び割り当て解除方法を提供することが必要とされる。

【0008】

【目的】上記背景に鑑み、本発明の目的は、コンピュータにおいて呼び出し及び復帰操作の平均速度を向上させることにある。他の目的は、プロセッサの処理においてレジスタセーブ及び復元の回数を最小限にすることにある。他の目的は、呼び出されたプロシージャにより必要とされる一時的なローカル記憶装置を効果的に割り当てることにある。更に本発明の目的は、ルーチンの呼び出し元、あるいは呼び出し先が使用する記憶装置にかかわらず十分なレジスタ記憶装置を割り当てることにある。更に他の目的は、現在使われていないレジスタについてはセーブや復元を行わないことによって、効率を向上することにある。他の目的は、プロセッサ内で使用可能な限定された個数のレジスタを割り当て及びセーブすることに関連するオーバーヘッドをへらすことにある。更に他の目的は、プロシージャが必要とするならばレジスタの全範囲をそのプロシージャにより利用できるようにすることまで含む、呼び出されたプロシージャの要求を満たすために、レジスタセットを動的に分割することにある。他の目的は、ソフトウェアのパイプライン化されたループのためにプロシージャが必要とするのと丁度同じ個数の循環レジスタをそのプロシージャに割り当てることにある。他の目的は、レジスタの使用密度を増加させることにある。更に他の目的は、コンパイラの介入なしに

レジスタのセーブ及び復元を行うことにある。

【0009】

【概要】本発明の1つの側面は、物理レジスタをスタティックレジスタとスタックレジスタに分割することにある。これにより、スタックレジスタがスタックをポインティングするベースレジスタまたはリロケーションレジスタを介して間接的にアドレスできるようになる。プロシージャ呼び出しの際にレジスタをセーブしプロシージャからの復帰の際にセーブされていたレジスタを復元するようにプロシージャに要請する代わりに、本方法では、全てのプロシージャが、その呼び出し元とは独立した一組のレジスタをスタックから割り当てる（プロシージャからの復帰の際にはその様なレジスタの割り当てを解除してスタックへ戻す）ことができるようにする。もしそのような割り当てがスタックオーバーフローまたはアンダーフローをもたらさないなら、メモリアクセスは不要である。

【0010】もしハードウェアが十分大きなスタックを実現するなら、呼び出されたプロシージャがローカルレジスタを直ちに入手できること、それがない場合にはレジスタのセーブ及び復元を必要とするメモリパイプが使えること、及びメモリとのトラフィックが減少することからもたらされるキャッシュ動作の改善により、システムスループット、資源利用及びプログラム実行時間の改善が期待される。

【0011】プロシージャにより要求される（また、おそらくは必要な）レジスタの正確な個数とそのプロシージャに割り当てられる。もっと具体的には、本発明によれば、各プロシージャ及び各グループには正確にそれら特性に合う所要の個数のレジスタが割り当てられる。このようにして、レジスタは不必要に割り当てられることもなく不必要に保持／復帰されることもない。この特徴はレジスタの効果的な使用及び実行時間の短縮につながる。

【0012】本発明はコンパイラの介入なしにディジタルコンピュータで動的にレジスタをプロシージャに割り当てる方法を含む。この方法は以下のステップを含む：複数のスタックレジスタを有する論理レジスタスタックを定義するステップ；ローカルリロケーション項（local relocation term, "l r e l" と称する）を初期化して、前記論理レジスタスタックをコンピュータの前記物理レジスタセットにマッピングするためのオフセットを定義するステップ；第1のスタックポインタ値（TOL）を初期化することにより、第1のプロシージャにより指定される任意の個数のスタックレジスタをローカルレジスタとして第1のプロシージャに割り当てて、前記論理レジスタスタック内の前記ローカルレジスタの範囲を区切るステップ；前記第1のプロシージャの実行の間の間のレジスタアクセス操作について、各ローカルレジスタを前記ローカルリロケーション項に応じて前記物理

レジスタセットにマッピングするステップ。

【0013】第2のプロシージャを呼び出すための準備にあたっては、この方法は以下のステップを要する；第1のスタックポインタ値（TOL）を“以前のTOL”（OTOL）と称される第2のスタックポインタ値としてストアするステップ；第1のスタックポインタ値（TOL）をインクリメントすることにより、第1のプロシージャにより指定された個数のスタックレジスタを第1のプロシージャにパラメータ引渡しレジスタとして追加して割り当てて、パラメータ引渡しレジスタを含めるようにするステップ；呼び出されたプロシージャが参照するように、割り当てられたパラメータ引渡しレジスタ内に選択されたパラメータをストアするステップ。我々はまた、パラメータ引渡しレジスタをローカルリロケーション項に応じて物理レジスタセットにマッピングする。

【0014】第2のプロシージャを呼び出すに当たって、この方法は更に第1のプロシージャパラメータ引渡しレジスタを含む初期ローカルレジスタ空間を第2のプロシージャに割り当てるステップを含む。このステップにより、これらのレジスタにストアされたパラメータを、メモリを参照せずに第2のプロシージャから利用できるようにする。第2のプロシージャが必要とする個数の追加のスタックレジスタは、スタックポインタ値をインクリメントすることによりローカルレジスタとして第2のプロシージャに割り当てられる。この割り当ては、第1のプロシージャのローカルレジスタ内容を先ずメモリにセーブするという操作を経ずに実行される。第2のプロシージャからの復帰に当たっては、この方法では、スタックポインタ値をローカルレジスタの個数分デクリメントすることにより、ローカルレジスタの割り当てを解除する操作が行われる。このように、本発明には、ローカルレジスタ内容をセーブ及び復元することなしに第2のプロシージャを呼び出したそこから復帰することが含まれる。

【0015】本発明の他の側面は、物理アドレスをレジスタファイルポートに提供するレジスタファイルポートアクセス回路である。この回路は仮想アドレスを受け取り、仮想アドレスをスタティックレジスタアドレス空間と比較してこの仮想アドレスがスタティックレジスタアドレス空間内にあるか否かを提示する。もしスタティックレジスタアドレス空間内にあれば、この回路はこの仮想アドレスを、対応レジスタにアクセスするための第1の物理アドレスとしてレジスタファイルポートに結合する。この回路には更に、受け取った仮想アドレスをローカルリロケーション項と結合して第2の物理アドレスを形成する回路と、仮想アドレスがスタティックレジスタアドレス空間内になければ第2の物理アドレスをアドレスとしてレジスタファイルポートに結合する手段も含まれる。アクセス回路は、ローカルリロケーション項を仮想アドレスと予め定められた物理レジスタの総数を法(m

odulo)とするモジュロ加算を行うように構成される。

【0016】本発明の、上述した、またそれ以外の目的、特徴及び利点は、以下の図面を参照して進められる好適な実施例の詳細な説明からもっと容易に明らかになるであろう。

【0017】

【実施例】図1はレジスタウインドウとして知られている、レジスタを割り当てる従来技術の方法を示す概念図である。以下において、参照番号は、図示されているアドレス空間モデルを参照するために使用される。図の参照番号をレジスタ番号と混同してはいけない。ここでは、物理レジスタ番号を示すために小文字のrを使用し、論理または仮想レジスタスタック番号を示すために大文字のRを使用する。省略記法“VR”は仮想レジスタを意味し、“PR”は物理レジスタを意味する。

【0018】図1において、番号n-1がついた第1のウインドウには、グローバルレジスタr0からr9及びローカルレジスタR10からR31が割り当てられる。新たなプロシージャが呼び出されると、別の一連のレジスタがこの新たなプロシージャに割り当てられる。番号1のウインドウを参照すると、レジスタr0からr9は、グローバルであるので、そのままである。呼び出しの後、6個のレジスタが前段のウインドウにオーバーラップして、呼び出し側のレジスタR10からR15までがレジスタR31からR26になる。10個のレジスタはウインドウに含まれないので、各プロシージャからは一度に32個のレジスタが見えていても、ウインドウ毎に16個(32-10-6)の固有のレジスタがある。オーバーラップしたレジスタはパラメータ引渡のために使用される。同様に、ウインドウ番号n+1において、呼出側のレジスタのR10からR15(ウインドウn)は、呼び出し後、R31からR26となり、再び、6個の重複レジスタが提供される。従来技術で述べたように、固定サイズでの分割を行うレジスタウインドウ技術は、使用されないときでもセーブされるレジスタを生成する。

【0019】本発明は、個々のプロシージャが物理レジスタのプールからあるいはそこへの任意の個数(論理レジスタスタックの数により制限される)のレジスタの割り当て及び割り当て解除を必要に応じて実行できるようにすることにより、プロシージャインターフェースにおいてセーブされた復元されるレジスタの個数を最小限にする。このプールの中のレジスタは、以下に述べる間接あるいはリロケーションポインタを介してアクセスされる。レジスタスタックはレジスタで利用可能なソフトウェアスタックの頂部として見ることができ、従って、容易にかつ速やかにアクセスされる。そのようなレジスタの“動的割り当て(dynamic allocation)”は、コンパイラにより予め定められているのではなく、呼び出されたプロシージャ自身により制御される。

【0020】本発明の一実施例では、128個の固定小数点及び128個の浮動小数点レジスタを設けてよい。典型的なハードウェアレジスタファイルは、64個のスタティックレジスタ及び64個の循環レジスタを有している。レジスタファイルは独立した集積回路で実現してよいし、あるいはプロセッサデバイス上に実装してよい。物理レジスタファイル自体の実現の詳細は知られており、ここではあまり関係がない。本発明は固定小数点レジスタと浮動小数点レジスタの一方または両方に等しく適用可能である。説明の都合上、固定小数点レジスタに適用するものとして本発明を説明する。以下の説明では、次に示す用語を使用する。

【0021】物理レジスタ(PR): システムアーキテクチャから見える物理レジスタ。物理レジスタの実際の個数は単なる設計事項である。物理レジスタはレジスタファイル内で実現されると仮定している。

仮想レジスタ(VR): 命令中で指定されるレジスタ番号である。VR番号はPR番号と同じであってよいし、あるいはVRを修正して、対応するPRのアドレスを決定してもよい(以下で説明する)。

スタティックレジスタ: グローバルレジスタ(Global Registers)とも称され、スタック操作あるいは循環に関与しないレジスタである。言い換えれば、これらレジスタのは、間接操作なしで、シラブル(syllable)で提供されるレジスタアドレスを使用して直接アクセスされる。以下の実施例では、VRアドレス0から31は、スタティックレジスタであるPR0から31にアクセスするために修正なしに使用される。

循環レジスタ(Rotating Registers): これらのレジスタはソフトウェアパイプライン化に関与するための、プロシージャにより割り当てられたレジスタであり、循環レジスタベースRRB(Rotating Register Base)からのオフセットとしてアクセスされる。どのプロシージャも、いかなる時でも物理レジスタの個数によりほぼ制限される任意の個数の循環レジスタにアクセスできる。循環レジスタはリングとしてアドレスされる。

スタックレジスタ: スタック操作及び循環に関与するレジスタのプールである。スタックレジスタをアクセスするためのベースあるいは間接操作を指定するスタックポインタを使うことにより、スタックレジスタのプールはリングとして管理される。(従って、循環レジスタはリング内のリングとして管理される。)言い換えれば、もしVR(i)が物理的に実現されているスタックレジスタの中の最大のアドレスを有しているものに対応するならば、VR(i+1)は最小アドレスを有するスタックレジスタに対応する。以下に述べる実施例では、96個のスタックレジスタ(R32からR127)が存在する。循環レジスタはスタックレジスタのプールから抜き出される。VRs32から127は、ベースポインタにより修正されて、対応するPRを決定する。

ローカルレジスタ(Local Registers): 現プロシージャからアクセス可能なスタックレジスタである。

【0022】 上述の循環レジスタベース(RRB)に加えて、好ましい実施例はスタックレジスタファイルの中をポインティングする次のような追加の間接つまりベースレジスタを維持する。(好ましくは、プロシージャは次のベースレジスタの各々の2つのコピーを有する。その1つは固定小数点スタックであり、他方は浮動小数点スタックである。)

有効レジスタ群の下限(底)BOV: レジスタスタックを介してアクセス可能なソフトウェアスタックの深さをマーキングするスタックポインタである。BOVを越えて割り当て処理を実行するとスタックのオーバフローを招き、BOVを越えて割り当て解除処理を実行するとスタックのアンダフローを招く。以下で更に説明する。

ローカルレジスタ群の下限BOL: 現プロシージャからアクセス可能なスタックレジスタ群の一方の端の境界を示すスタックマーカである。現プロシージャの全てのスタックレジスタはBOLに関してアクセスされる。一般に、Pがスタックレジスタの総数であり、i及びjが、 $i < 32$ の場合 $j = i$ 、 $i \geq 32$ の場合 $j = [(BOL + i - 32) \bmod P] + 32$ によって関連付けられるとすると、VRiはPRjにアクセスする(ここで、PR0からPR31はスタティックレジスタであると仮定している)。好ましい実施例において、BOLはデフォルトで最初のスタックレジスタ(PR32)を指す。

ローカルレジスタ群の上限(頂上)TOL: 現プロシージャからアクセス可能なレジスタ群の他端の境界を示すスタックマーカである。プロシージャがローカルエリア内に存在しないレジスタ、すなわちBOL境界からTOL境界までの範囲の外にあるレジスタにアクセスしようとするると例外を引き起こす。

ローカルレジスタ群の以前の上限OTOL: パラメータレジスタの割り当ての行われる以前のTOLの値である。TOLとOTOLの間の一連のレジスタは割り当てられたパラメータレジスタである。

循環レジスタ群の下限BOR: 循環にかかわるスタックレジスタ群の一端の境界を示すスタックマーカである。循環レジスタ群の上限TOR: 循環にかかわるスタックレジスタ群の他端の境界を示すスタックマーカである。

【0023】 一般に、レジスタスタックにおける利益はスタックレジスタにだけあてはまり、コンパイラは依然としてスタティックレジスタのための呼び出し元/呼び出し先、セーブ/復元戦略を採り入れ続けねばならない。全ての議論及び説明は固定小数点スタックと浮動小数点スタックに等しく適用される。各スタックは自分自身のベースレジスタセットを有する。固定小数点レジスタファイル及び浮動小数点レジスタファイルは各々別々に制御される。本発明を説明するために、固定小数点ス

タックを詳細に述べる。

【0024】 ローカル、パラメータ及び循環レジスタは新たに定義された操作であるallocを実行することにより割り当てられまたは割り当て解除される。ローカルレジスタが割り当てられ/割り当て解除される場合には、レジスタの割り当て及び割り当て解除により、TOL及びOTOLが修正される。TOL及びOTOLは割り当て及び割り当て解除されるローカルレジスタの個数だけインクリメント/デクリメントされる。パラメータレジスタの割り当て/割り当て解除により、TOLが修正される。TOLは、以下の図3A-Iに示されるように、割り当て及び割り当て解除されるパラメータレジスタの個数だけインクリメント/デクリメントされる。レジスタの割り当て及び割り当て解除は、スタックがオーバフロー/アンダフローするときBOVにも影響を及ぼすかもしれない。

【0025】 循環レジスタの割り当て/割り当て解除により、BOR、TOR、TOL、及びOTOLが修正される。循環レジスタが割り当てられる際、BORはTOLの内容にセットされる。TOL、OTOL及びTORはTOLと割り当てられる循環レジスタの個数の合計にセットされる。循環レジスタの割り当て解除より、逆方向の修正が行われる。上述した機構により、プログラムに影響を与えることなしに、例えば装置モデル毎に物理レジスタの個数を変えることができる。

【0026】 プロシージャ呼出し(すなわち、ブランドリンクの実行)の実行に当たって、各種のベースレジスタの現在の状態がパラメータレジスタ0にストアされる。よってコンパイラは呼び出されたプロシージャへ渡す/呼び出されたプロシージャから送られるパラメータの個数以外に、追加のパラメータレジスタを1個割り当てなければならぬ。更にBOLはOTOLの値にセットされ、OTOLはTOLの値にセットされる。プロシージャからの復帰に当たっては、各種のベースレジスタはパラメータレジスタ0にストアされた値に設定しなおされる。プロシージャからの復帰は、呼び出されたプロシージャのローカルレジスタ及び循環レジスタの割り当て解除を伴うので、スタックアンダフローを招くことがある。

【0027】 割り当てを行っている際にTOLがBOVを越えようとするとき、スタックはオーバフローしたと言われる。全ての演算が、スタックに実装されているレジスタの個数を法として行われるということを想起されたい。あるいは、モジュロ加算(modulo-plus)機能を使用して固定小数点レジスタアドレス空間を飛び越してもよい。同様に、割り当て解除を行っている際にBOLがBOVを越えようとするとき、スタックはアンダフローしたと言われる。オーバフロー/アンダフローの発生はハードウェアで検出され、トラップハンドラが適切にスタックレジスタをソフトウェアスタックに追出

すためにソフトウェアスタックからスタックレジスタへの復元を行うために呼び出される。

【0028】上述した機構により、スタックオーバーフロー及びアンダーフローを予測してレジスタスタックの内容の流し出したそこへの充填をバックグラウンドで行うハードウェア（ソフトウェア）を使用することができる。従って、従来の意味でのスタックオーバーフロー及びアンダーフローは、後述するところの我々がレジスタ清浄化(register cleaning)と呼んでいる処理により回避することができる。

【0029】割り当て処理操作は、例を使うことによって最も良く説明できる。プロシージャAでBOLが物理レジスタ38をポインティングし、TOLが物理レジスタ47をポインティングすると仮定する。するとプロシージャAは10個のローカルレジスタを有する。プロシージャBの呼び出しに先立って、プロシージャAは4個のパラメータレジスタを割り当てる。これにより、OTOLが47にセットされ、TOLは51にセットされる。プランチアンドリンクが実行されると、これらのベースレジスタの値がパラメータレジスタ0、すなわち物理レジスタ48にバックされる（以下の制御レジスタAについての説明を参照）。更に、BOLは47にセットされる。呼び出されたプロシージャ（B）のローカル空間の下限をOTOLに配置することにより、パラメータレジスタは両者に共通となり、またこれらのレジスタ部分はBのローカル領域の下部となる。OTOL及びTOLは51にセットされる。プロシージャBに10個のローカルレジスタを割り当てると仮定する。この場合、TOL及びOTOLを60にセットする。プロシージャBから復帰すると、BOL、TOL及びOTOLは夫々初期値38、47、51にリセットされる。

【0030】ある実施例では、プロシージャBによりプロシージャAへ戻される値を物理レジスタ48から51に入れることができる。その代わりに、復帰値をスタティックレジスタに入れることもできる。これにより、復帰後直ちにパラメータレジスタの割り当てを解除して次のプロシージャ呼び出しのために利用できる。

【0031】プロシージャが呼び出されると1つのパラメータレジスタ（ここでは48）を使ってポインタ値がストアされるということにも注意されたい。もっと具体的に言えば、復帰情報は制御レジスタに記憶され（以下で更に説明する）、コンパイラは、復帰前に、復帰情報をローカルレジスタ領域にコピーし、また制御レジスタにストアすることが必要とされる。好ましくは、TOL及びOTOL値自体はセーブされないが、それらの値を計算できるようにする代わりの値、公称上、以前の値に対するオフセット、がセーブされる。オフセット値を使用することによって、これらの値を任意のレジスタに記憶できるようになる。かくして、レジスタを任意量だけ循環しても、上記機構は依然として正確に動作する。

他の実施例では、この目的のためにパラメータレジスタを余分に割り当てて、パラメータ引渡しに使用できるレジスタの総数を呼び出し側のプロシージャにより割り当てられた実際の数に等しくなるようにする。循環レジスタの割り当て／割り当て解除は、同様の態様で動作する。

【0032】各レジスタスタックは、スタックオーバーフローの際にはレジスタがセーブされ、またスタックアンダーフローの際にはレジスタが読み出される固有のソフトウェアスタックを有する。従って、各レジスタスタックはまさに適当なソフトウェアの先端部分を表す。

【0033】上述の方法を実現するに当たって、以下の制御レジスタを提供するのが好都合である：

制御レジスタA：これは固定小数点スタック用の各種のベースポインタBOV、BOL、TOL、BOR、TOR及びOTOLを収容する。

制御レジスタB：これは浮動小数点用の別のベースポインタBOV、BOL、TOL、BOR、TOR及びOTOLを収容する。

制御レジスタC：これは固定小数点レジスタのためのレジスタスタックをバックアップしているソフトウェアスタックのメモリアドレスを含む。

制御レジスタD：これは浮動小数点レジスタのためのレジスタスタックをバックアップしているソフトウェアスタックのメモリアドレスを含む。

プロシージャ呼び出しの実行の準備として、適当なベースポインタがパラメータレジスタ0にストアされることを想起されたい。

【0034】図2に目を向けると、R0からR127までの番号が付与されたレジスタセットについての論理アドレス空間モデルが示されている。スタティックレジスタ50（R0からR31）は例えばグローバル値のためにリザーブされ、ローカルレジスタ割り当て機構には関与しない。スタックレジスタR32からR127は参照番号58により示される（この参照番号はアドレスではないことに注意）。このモデルにより示される仮想アドレスはソフトウェアプロシージャから見たレジスタスタックを示す。仮想アドレス（VR）は、以下に述べられる物理レジスタファイルにアクセスするために実際のつまり物理レジスタアドレス（PR）に変換される。初期状態では、割り当てられていないアドレス空間60がレジスタスタック全体を構成している。

【0035】図3Aから図3Iは、呼び出された一連のプロシージャから見える仮想アドレス空間を示している。呼び出されたプロシージャは、各図の上部に書かれたA、B、C、Dで示している。このモデルで方向「上」、「下」、及び表記「上限」、「下限」は任意である。例えば、ローカルレジスタを「上限」、ここではR127から下向きに割り当て、スタックの下限（VR32）に到達したとき回り込みを行うこともできる。我々は、R32から上向きに割り当てることによって本発

明を説明する方を選択した。動作の原理は、一貫性を保つ限り同一である。

【0036】図3Aに目を向けると、第1のプロシージャAへの呼び出しを行った後の図2の論理アドレス空間が示されている。論理アドレス空間（すなわち一連の仮想レジスタ）62はプロシージャAに対してローカルなものとしてプロシージャAに割り当てられている。BOL (Bottom of Local) ポインタはプロシージャAのローカル空間の下限を示し、TOL (Top of Local) はプロシージャAのローカルアドレス空間の上限を区切っている。BOV (Bottom of Valid) は、BOLの値に初期化され、現在割り当てられている空間の範囲を区切っている。参照番号60は未だ割り当てられていない仮想レジスタ（つまりアドレス空間）、すなわちTOLよりも上の空間またはBOVよりも下の空間を示す。図3Bにおいて、プロシージャAは次に呼び出されるプロシージャへのパラメータ引渡のためにパラメータ空間64を割り当てる。パラメータ空間64はプロシージャAに割り当てられているローカルアドレス空間をインクリメントする。この様子は、それに合わせてTOLポインタがパラメータ空間64の上限に上向きに調整されていることに示されている。ポインタOTOLはパラメータレジスタの割り当ての前のTOL値を示している。

【0037】プロシージャAは次にプロシージャBを呼び出す。注意したように、ポインタ（制御レジスタA）は第1のパラメータレジスタにストアされる。図3Cを参照すると、いつものように、プロシージャBは、BOLポインタで表わされるように、スタックの下限（VR33）で始まるローカル（仮想）アドレス空間を割り当てる。プロシージャBのローカル空間の最初の部分は、パラメータ空間64がプロシージャAとBに共通になるように、プロシージャAのパラメータ引渡レジスタ64にマッピングされる。呼び出されたプロシージャのローカル空間は常にスタックの下限（BOL）で始まり、この空間はまた呼び出し側のプロシージャのパラメータ空間で始まる。

【0038】従って、プロシージャ呼び出しは、パラメータ引渡空間（例えば64）が下限側に向かうように仮想レジスタスタックを押し込む、つまりプッシュダウンすると考えることができる。呼び出し側プロシージャのパラメータ空間（例えば62）は、図3Cのアドレス空間の上限部分に回り込み、BOV (Bottom of Valid) ポインタを調整することにより区切られている。プロシージャBも（純粋にローカルな）レジスタ66を更に割り当て、TOLポインタにより区切っている。上と同様に、残りの割り当てられずに残っているアドレス空間は60によって示される。

【0039】図4Aから図4Iは、レジスタファイルのような物理アドレス空間をモデル化している。この段階で、図3Aから図3Iにモデル化された仮想アドレス空

間と物理アドレス空間の関係を質的に考慮することは役に立つ。図4Aを参照すると、BOL及びBOVポインタはレジスタファイルアドレス空間の原点を示している。これらは例えば物理アドレス0で良い。プロシージャAの仮想アドレス空間62（図3A）はTOLポインタで区切られている物理アドレス空間102（図4A）に対応する。図4Bはまた、プロシージャAにより割り当てられ、図3Bの仮想アドレス空間64に対応するパラメータ空間104も示している。参照番号100は物理アドレス空間モデル中で現在割り当てられていないアドレス空間を示している。一般に、図3Aから図3Iの参照番号に40を加算すれば、夫々図4Aから図4Iの対応する参照番号になる。

【0040】図4Cは、TOLポインタを調整することにより図3CのプロシージャBのローカルアドレス空間66に対応するアドレス空間106が更に割り当てられたことを示している。従って、呼び出されたプロシージャの仮想アドレス空間は常にレジスタスタックの下限から始まるが、物理レジスタファイル内ではこれに対応するデータのリロケーションはないことが観察できる。リロケーションを実際に行う代わりに、図4Aから図4Iに示されるように、追加されるレジスタは、前に割り当てられた物理アドレス空間に影響を与えることなしに呼び出されたプロシージャにより必要に応じて割り当てられる。次に、これ以上のプロシージャ呼び出しを考慮するために図3Dを参照する。

【0041】プロシージャBは図3Dに示されるように、TOLポインタを調整することによりパラメータ引渡アドレス空間70を割り当てる。プロシージャAと共通のパラメータアドレス空間64を有する残りのローカルアドレス空間66は影響されない。図3Dに示され、BOVアドレスポインタによって表わされるプロシージャAのローカルアドレス空間62は、アドレス空間の上限部にそのまま残っている。

【0042】図3Eを参照すると、プロシージャBは更に別のプロシージャCを呼び出す。プロシージャCについての論理アドレス空間は以下の部分で構成される：論理アドレス空間の下限（BOL）で始まり、アドレス空間70はプロシージャBと共通のパラメータ引渡空間である。プロシージャCはTOLポインタで表わされるローカルアドレス空間72を割り当てる。呼び出し側プロシージャ（B）のローカルアドレス空間64、66は、プッシュダウンされ、図3Eのモデルの上限部分に回り込む。プロシージャAローカル空間62は現在の呼び出しに適応するためにプッシュダウンされ、またBOVがそれに従って動かされる。言い換えれば、レジスタスタックは論理的に循環する。いつものように、残りの割り当てられていないアドレス空間は60で示される。

【0043】図4D及び図4Eは夫々図3D及び図3Eでモデル化された論理アドレス空間に対応する物理アド

レス空間を示している。図4Dを参照すると、パラメータ引渡空間110は、図3DでプロシージャBにより割り当てられた仮想パラメータ空間70に対応する。同様に、図4Eのローカルアドレス空間112は、図3EでプロシージャCにより割り当てられた仮想ローカルアドレス空間72に対応する。

【0044】ここで図3Fを参照すると、プロシージャCは前に割り当てられたローカル空間72に加えて循環レジスタアドレス空間74を割り当てる。TOL（及びTOR；図3G参照）ポインタは循環レジスタ空間の上限を示し、BORは循環レジスタ空間の下限を示している。仮想アドレス空間62、64及び66は影響されない。図4Fは、BOR及びTOLで境界付けられる物理アドレス空間114の対応する割り当てを示している。循環レジスタの個数は、ソフトウェアのパイプライン化されたループの特性に応じて変えられる。プロシージャが現在利用可能なアドレス空間を越える循環レジスタ空間を割り当てようとする場合のみ、レジスタオーバーフローが起こる。この場合については、以下に述べられる。

【0045】次に、図3Gを参照すると、他のプロシージャの呼び出しを見越してプロシージャCは循環レジスタ空間74の上限の上にパラメータ引渡空間76を割り当てる。パラメータ空間を区切るためにTOLを調整する。論理アドレス空間60はまだ割り当てられていないレジスタである。図4Gは、プロシージャCが他のプロシージャにパラメータを通すための物理アドレス空間116の対応する割り当てを示している。

【0046】図3Hはもう1つのプロシージャDがプロシージャCにより呼び出された後の仮想アドレス空間モデルを示している。前にプロシージャCにより割り当てられたパラメータ空間76は、いつものように、プロシージャDのローカルアドレス空間の下限に現れている。更に、プロシージャDはTOLポインタを調整することによりローカルアドレス空間78を割り当てる。呼び出しプロシージャ、すなわちプロシージャC、についてローカルなアドレス空間（共通パラメータ引渡空間76を除く）は、図3Hの70、72及び74で示されるように、プッシュダウンされ、モデルの上限部に回り込んでいる。プロシージャBについてローカルなアドレス空間66及び64はこれに従ってプッシュダウンされる。同様に、プロシージャAのローカル空間62は、スタック上で更にプッシュダウンされ、未だ以って割り当てられていないアドレス空間60を残して、BOV(Bottom of Valid)ポインタにより区切られる。図4Hは、プロシージャDがローカルレジスタとして使用する物理アドレス空間118の対応する割り当てを示す。

【0047】プロシージャDは次に、図3Hの60により示される利用可能なアドレス空間を越えて循環レジスタ空間を割り当てようとする。これはレジスタオーバー

フロー条件を引き起こす。その結果、BOVポインタの上方にあるメモリ空間の部分がメモリ（図示せず）にセーブされる。セーブされる部分は論理アドレス空間62と64及び66の一部分を含む。BOVポインタはオーバーフローセーブ操作の結果繰り上がり、これにより空いている空間を追加する。この結果の割り当てられていないアドレス空間60は、プロシージャDの循環レジスタの要求に合致するよりも大きい。この結果は図3Iに示されており、ここで80はプロシージャDの循環レジスタ空間を示している。

【0048】図3Iを参照すると、プロシージャDはBOR及びTOLポインタで区切られている循環レジスタアドレス空間80を割り当てし終わっている。この場合、必要最小限の空間より幾分か大きいものがメモリにセーブされている。その結果、割り当てられていない部分60が残っている。これは単に最小限の今すぐに必要というのではなく、予め定められた個数のアドレスを動かすようにオーバーフローセーブ機構を構成していることから起こる。上述のセーブ操作によってリロケーションされたアドレスの個数は、好ましくは従属ハードウェアで効率よく実現できるように選択される。この結果現れるヒステリシスは、使用時に必要になるメモリ参照の回数を減少させ得る。他の実施例では、保留されている割り当てを調整するために十分なアドレス空間のみをセーブする。レジスタオーバーフロー及びセーブ後であって、所要の循環レジスタを割り当てた後の物理アドレスモデルは図4Iに示される。ここで120はプロシージャDの循環レジスタ空間を示している。

【0049】レジスタオーバーフローセーブ及び復元機構の詳細は知られている。しかし、本発明の他の側面は、完全にレジスタオーバーフローを回避するための、仮想アドレスレジスタシステムとともに作動する「清浄化(cleaning)」機構にある。「清浄な(clean)」レジスタは、メモリ内容の正確なコピーを有するレジスタとして定義される。逆に、「汚染した(dirty)」レジスタはメモリ内容の信頼できるコピーを持っていない。汚染したレジスタは恐らくは有効なものである。すなわち、おそらく現在割り当てられている。清浄なレジスタ空間はBOC(Bottom of Clean)及びTOC(Top of Clean)ポインタで表わされている。BOCは本質的にはBOVと同じである。初期状態では、定義により、レジスタ内容がメモリにコピーされるまでは清浄なレジスタが存在しないので、TOCはBOCに等しい。レジスタの清浄化はバックグラウンドにおいてトランスペアレントに、すなわちもしそうでなければ遊んでしまうプロセッササイクルを「盗む」ことにより実行される。

【0050】TOCがBOLより小さいとき、あるレジスタはメモリをまだ更新していない。レジスタ清浄化機構は次のレジスタ、すなわちTOC+1での値をメモリにコピーする。この機構は次にTOCをインクリメント

し、その結果、TOCは常に先頭の清浄なレジスタをポインティングする。一般にローカルレジスタは頻繁に汚染されるので、無視することができる。そこでBOLまでしか清浄化しないことが好ましい。清浄化操作はソフトウェアからはトランスペアレントであり、ここで説明するレジスタ割り当て／割り当て解除方法及び装置とは独立である。

【0051】レジスタファイルポートアクセス回路

ケース1：スタティックレジスタアクセス

図5は本発明に関するレジスタファイルポートアクセス回路140のブロック図を示している。物理レジスタ、例えば128個のレジスタは、レジスタファイル144のような一連のハードウェアレジスタファイル中に提供される。図5に示されるタイプのアクセス回路が各レジスタファイルポートに対して提供される。回路機能の1つは、例えばソフトウェアプロシージャにより提供されるアドレスである論理アドレスRを、レジスタファイルにアクセスするための対応する物理レジスタアドレスにマッピングすることである。回路140において、論理レジスタアドレスRがライン142上に入力され、マルチプレクサ146への3つの入力の1つに結合される。

【0052】コンパレータ150は、Rの値を具体的なアプリケーション中でのグローバルつまりスタティックレジスタの個数（この例では32）に等しい定数と比較し、この論理アドレスがスタティックレジスタの範囲の中に入るかどうか（つまり、 $R < 32$ かどうか）を判定する。もしRが32より小さいときは、提示されたアドレスがスタティックレジスタの範囲内にあり、コンパレータ150からの出力はマルチプレクサ制御ライン152をアサートして、MUX146が物理アドレスとしてレジスタファイルに入力するものとして値R自体を選択するようにする。言い換えれば、Rはスタティックレジスタについては修正されない。上述のように、スタティックレジスタはスタックレジスタ操作には関与しない。

【0053】ケース2：レジスタスタックアクセス

もしRが32以上（かつqupまたはpdnがアサートされていない）ならば、Rは有効なレジスタスタック仮想アドレスであり、物理レジスタアドレスにマッピングされなければならない。さしあたり、循環レジスタが現プロシージャに割り当てられていないと仮定する。この場合、モジュロ加算を行う加算器(modulo-plus adder)154の出力はMUX146を介してレジスタファイル144に供給される物理アドレスとして選択される。モジュロ加算を行う加算器154は、物理レジスタアドレスを決定するためにモジュロ演算を使用して、論理アドレスRをローカルリロケーション項（“lrel”）、つまりオフセットと結合する。リロケーションの加算は、スタック内に物理的に実装されているレジスタ数についての法を取って実行される。ローカルリロケーション項lrelは、(BOL値) - (固定されたレ

ジスタの個数)に等しい。lrelは任意であり、予め定められたリロケーションオフセット量またはブロックサイズに限定されない、ということに注意されたい。このように、所与のプロシージャにより割り当てられた個数丁度のレジスタが使用される。逆に、復帰に当たっては、丁度同じ個数のレジスタが割り当て解除される。

【0054】説明のため、ハードウェアレジスタの総数が128個で、レジスタ0から31が固定されたレジスタであり、レジスタスタックが96個のレジスタを備えていると仮定する。次に、仮想スタックアドレスR=44及びBOL=40であると仮定する。すると、Rと(BOL-32)をモジュロ加算したものは $(44+8) \bmod 96 = 52$

である。この例では、モジュロ加算による回り込み（ラップアラウンド）は起こらない。しかし、もしBOL=90なら、

Rと(BOL-32)のモジュロ加算 $= (44+58) \bmod 96 = 6$

である。ただし、このモジュロ加算演算の結果は更にR0からR31を飛び越えるようにさせるので、最終的には38となる。一般には、Pをスタックレジスタの総数とし、iとjが、 $i < 32$ なら $j = i$ 、 $i \geq 32$ なら $j = [(BOL + i - 32) \bmod P] + 32$ という関係にあるとき、VRiはPRjをアクセスする。

【0055】ケース3：レジスタ復元及び清浄化
レジスタファイルポートアクセス回路140はまた、レジスタ復元及びレジスタ「清浄化」(register cleaning)のためにアクセスができるようにする。制御信号qupは、すでに書きされているがレジスタファイル内で再度有効(valid)にしなければならないレジスタを復元するためにメインメモリから読み出すことを指示する。この制御信号は、もっと多くの有効なレジスタを提供するためにスタックアンダーフローにともなって使用される。qupがアサートされると、この信号はMUX146を制御して、レジスタファイルにアクセスするためのアドレスとしてQUPを選択する。QUPは復元されるべき次のレジスタのアドレス、すなわちBOV-1である。

【0056】QUPはローカル空間の外側を清浄にするための次のレジスタのアドレスである。これは次の使用可能な、つまり有効ではないレジスタである。よって、QUPアドレスは単にTOL+1である。メインメモリの内容はレジスタファイルのこのアドレスにコピーされ、定義によりそのレジスタを清浄にする。TOCは常に清浄な空間の上限をポインティングするようにインクリメントされる。

【0057】制御信号qdnはメインメモリにその内容をコピー（書き込み）することによりレジスタを清浄にすることを指示する。qdnがアサートされると、それはQDNをレジスタファイルにアクセスするためのアド

レスとして選択するようにMUXを制御する。QDNは清浄にすべき次のレジスタのアドレス、すなわちTOC+1である。レジスタ清浄化機構はこのレジスタの内容をメインメモリにコピーする。この機構は次にTOCをインクリメントし、TOCが清浄なレジスタ群のうちの上限にあるものを常にポインティングするようにする。

【0058】QUP及びQDNは互いに排他的である。ポートはこれらの中の一つまたは他方の何れかを備えるが、両方ということはない。QUPはレジスタファイル内のストアポートで実現され、常にメモリからの読み出しを行う。QDNはレジスタファイル内の読み出しポートで実現され、常にメモリへの書き込みを行う。図5及び図6中の“QUPまたはQDN”という表記法は、図面を増やさずにこの相互の排他性を伝えようとするものである。

【0059】循環レジスタ実装付きのレジスタファイルポートアクセス

図6において、レジスタファイルポートアクセス回路160がブロックの形態で示されている。図6の回路160は図5の回路140と共通のいくつかの素子を備えている。ここで同じ様な参照番号は共通の回路素子を示す。共通の特徴の記載は省略する。図6はレジスタスタック内で循環レジスタを実現するための追加の回路素子を含んでいる。以前と同様に、論理アドレスRは入力ノード142に提供される。コンパレータ164は論理アドレスRとBOR (Bottom of Rotating) ポインタを比較する。別のコンパレータ166はRとTORポインタを比較する。RがBORより大きく、TORより小さいときは、論理アドレスは循環レジスタとして現プロシージャに割り当てられたレジスタを指している。

【0060】物理アドレスrはRに循環リロケーション項(rotating relocation term) rrelを加算したものに等しい。循環リロケーション項rrelは、循環レジスタ内で回り込みが起きないと仮定すると、循環レジスタセット内の循環を考慮に入れば、ローカルリロケーション項lrelに循環レジスタベース値(RRB)を加算したものに等しい。従って、

$$r = R + lrel + RRB$$

となる。しかし、循環レジスタ内で回り込みが起これば、

$$r = R + lrel + RRB - (TOR - BOR)$$

となる。

【0061】ここでTOR-BORは循環レジスタセットの大きさを与える。アクセス回路160で、Rは加算器170でrrelに加算され(前述のモジュール加算演算を使用)、その結果はMUX162に与えられる。

lrel及びRRBはRよりも前にわかるので、リロケーション項rrelを前以って計算しておくことができる。回り込みが起こる場合、別のリロケーション項rrelがモジュール加算器172でRに加算され、その

結果がMUX162に与えられる。ここで、rrelはlrel+RRB-(TOR-BOR)に等しい。rとしてMUX162で選択される値は、以下のように決定される。何も手を打たないと論理アドレスが循環レジスタの境界を越えてしまう場合に循環レジスタ内で回り込みが起こる。従って、問題は、

$$R - BOR + RRB > TOR - BOR$$

かどうか? ということである。代数学から、このテストはR > TOR - RRBかどうか? というテストと等価である。これは、RとTOR-RRBとを比較する、図6のコンパレータ168により判定される。その判定結果が真であれば、コンパレータ168は物理アドレスrとしてモジュールプラス加算器172の出力を選択するようにMUX162を制御する。結果が偽であれば、循環レジスタは回り込まなかったものであり、コンパレータ168は物理アドレスrとしてモジュールプラス加算器170の出力を選択するようにMUX162を制御する。

【0062】種々の回路は回路140または160の所要機能を達成するために工夫される。例えば、清浄なアドレスという特徴はあるアプリケーションでは実現されるが他のアプリケーションでは実現されない。あるアプリケーションは従属(subject)スタック内の循環レジスタを提供しないが、このようなアプリケーションでは図5のような回路で十分である。他のアプリケーションでは、循環オフセットRRBを他の場所で計算し、必要に応じて結果を加算器170に提供してよい。夫々の実現形態の、性能上のトレードオフを持ちやすい具体的内容は、本明細書を読んだ当業者には自明であろう。レジスタファイルポートアドレッシングがクリティカルパスであるアプリケーションでは、例えば高速の並列ハードウェアが提案される。

【0063】図7は本発明を実現するためのレジスタファイルシステムの一例を概略的に示すブロック図である。qup/pdn, TOR, BOR, RRB, QUP/QDN, lrel, rrel及びrrel#が付された一連のレジスタは、対応するポインタ値を維持するために提供される。これらのレジスタは、所要のポインタ値を、上述したレジスタファイルポートアクセス回路160のような再マッピング回路に供給するためにバス176を介して結合されている。レジスタファイル144内で使用される各レジスタファイルポート毎に、このような再マッピング回路が1つ提供される。この一般的な構成の種々の変形は、前述の目的及び動作を見れば、熟達したハードウェア設計者には明らかであろう。例えば、複数のポインタ値をもっと少ないレジスタ内に収めてもよい。選択された途中の値または途中のアドレスを、性能を最適化するために前以って計算しておいてもよい。種々の作業のハードウェアとソフトウェア(マイクロコードを含む)への振り分けのような他の変形は、

特定の実現形態についての設計上のトレードオフと適応の問題であり、これらは皆上記実施例と等価であると考えられる。

【0064】本発明の好ましい実施例で発明の原理を図示、説明したが、本発明はその原理から逸脱することなく配置及び細部を修正できることは当業者には明らかである。本願特許請求の範囲の精神及び範囲内で得られる全ての修正を本願の技術的範囲である。

【0065】

【効果】以上詳細に説明したように、本発明によればレジスタを有効に活用でき、プロシージャ呼出し／復帰に伴うレジスタセーブ／復元のためのメモリアクセスを最小化することができる方法及び装置が提供される。

【図面の簡単な説明】

【図1】レジスタウインドウを示す概念図。

【図2】一連のレジスタの論理アドレス空間モデルを示す図。

【図3 A】本発明の操作を示す図2のレジスタセットの論理アドレス空間モデルを示す図。

【図3 B】本発明の操作を示す図2のレジスタセットの論理アドレス空間モデルを示す図。

【図3 C】本発明の操作を示す図2のレジスタセットの論理アドレス空間モデルを示す図。

【図3 D】本発明の操作を示す図2のレジスタセットの論理アドレス空間モデルを示す図。

【図3 E】本発明の操作を示す図2のレジスタセットの論理アドレス空間モデルを示す図。

【図3 F】本発明の操作を示す図2のレジスタセットの論理アドレス空間モデルを示す図。

【図3 G】本発明の操作を示す図2のレジスタセットの論理アドレス空間モデルを示す図。

【図3 H】本発明の操作を示す図2のレジスタセットの論理アドレス空間モデルを示す図。

【図3 I】本発明の操作を示す図2のレジスタセットの論理アドレス空間モデルを示す図。

【図4 A】図3 Aの論理アドレス空間モデルに対応する

物理アドレス空間モデルを示す図。

【図4 B】図3 Bの論理アドレス空間モデルに対応する物理アドレス空間モデルを示す図。

【図4 C】図3 Cの論理アドレス空間モデルに対応する物理アドレス空間モデルを示す図。

【図4 D】図3 Dの論理アドレス空間モデルに対応する物理アドレス空間モデルを示す図。

【図4 E】図3 Eの論理アドレス空間モデルに対応する物理アドレス空間モデルを示す図。

10 【図4 F】図3 Fの論理アドレス空間モデルに対応する物理アドレス空間モデルを示す図。

【図4 G】図3 Gの論理アドレス空間モデルに対応する物理アドレス空間モデルを示す図。

【図4 H】図3 Hの論理アドレス空間モデルに対応する物理アドレス空間モデルを示す図。

【図4 I】図3 Iの論理アドレス空間モデルに対応する物理アドレス空間モデルを示す図。

【図5】本発明の一実施例を実現するレジスタファイルポートアクセス回路を示すハードウェアブロック図。

20 【図6】本発明の、レジスタスタック内に循環レジスタを有する他の実施例を実現するレジスタファイルポートアクセス回路を示すハードウェアブロック図。

【図7】本発明を実現するレジスタファイルシステムの一例を示すハードウェアブロック図。

【符号の説明】

BOV, BOL, TOL, OTOL, BOR, TOR :
ベースレジスタ

50 : スタティックレジスタ

60 : 割り当てられていないアドレス空間

30 140, 160 : レジスタファイルポートアクセス回路

144 : レジスタファイル

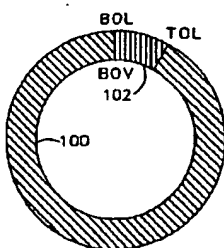
146 : マルチプレクサ

150, 164, 166, 168 : コンパレータ

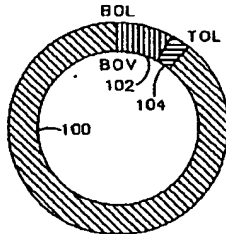
152 : マルチプレクサ制御ライン

154, 170, 172 : 加算器

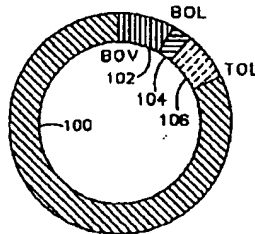
【図4 A】



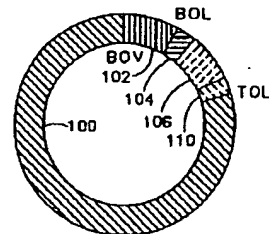
【図4 B】



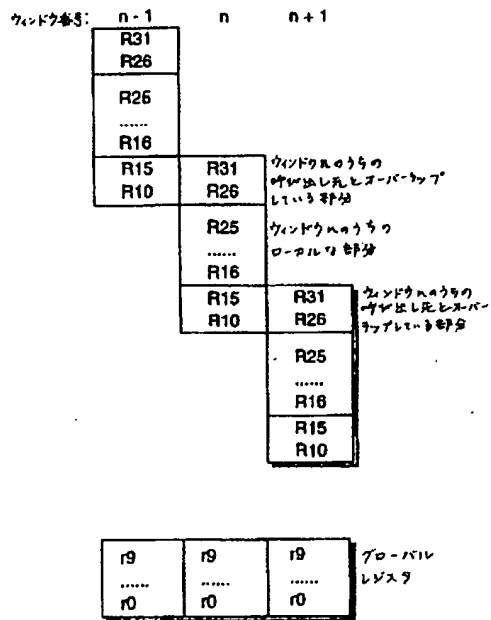
【図4 C】



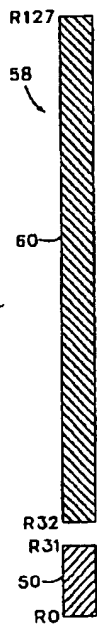
【図4 D】



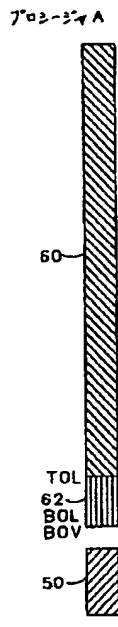
【図1】



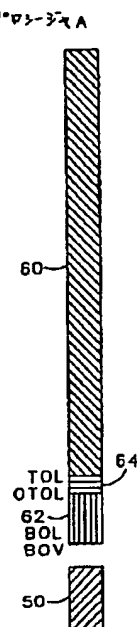
【図2】



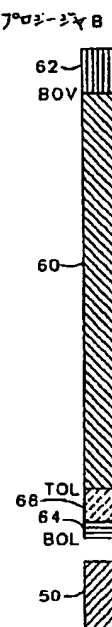
【図3A】



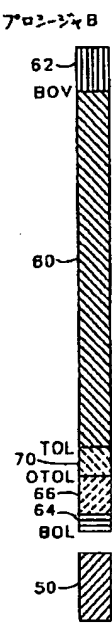
【図3B】



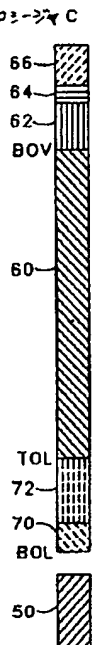
【図3C】



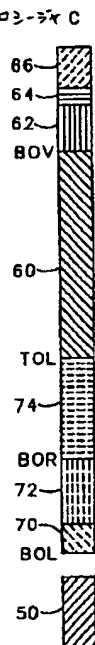
【図3D】



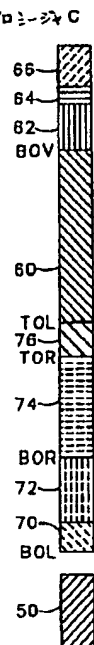
【図3E】



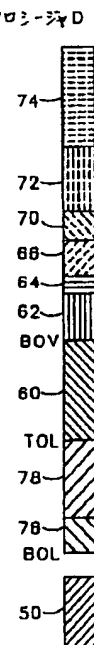
【図3F】



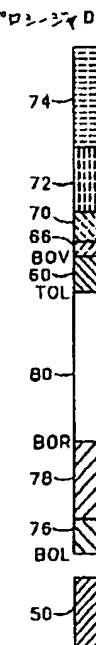
【図3G】



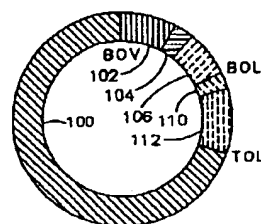
【図3H】



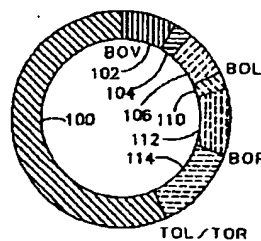
【図3I】



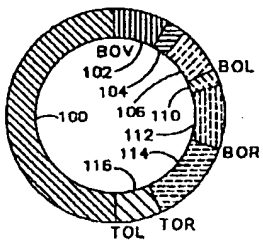
【図4E】



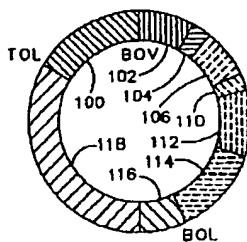
【図4F】



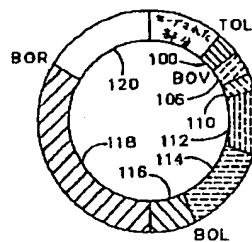
【図 4 G】



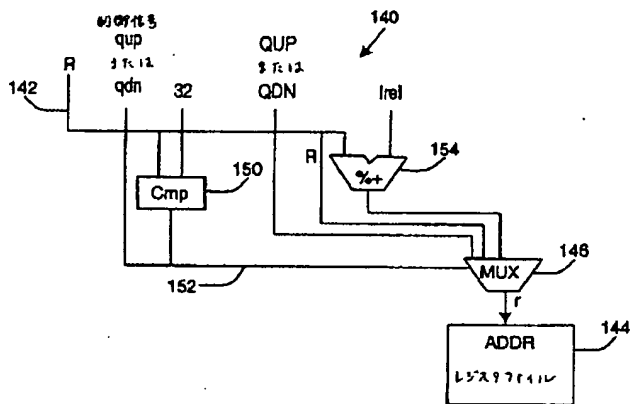
【図 4 H】



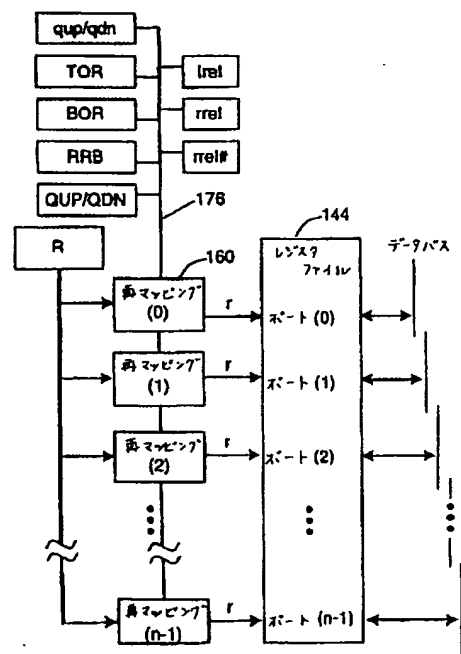
【図 4 I】



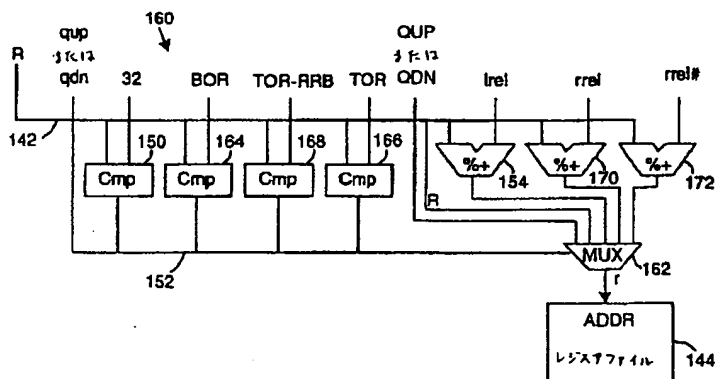
【図 5】



【図 7】



【図 6】



フロントページの続き

(72)発明者 ロバート・エム・イングリッシュ
アメリカ合衆国カリフォルニア州メンロ
ー・パーク、イースト・クリーク・プレイ
ス 4

(72)発明者 ラジブ・グプタ
アメリカ合衆国カリフォルニア州ロス・ア
ルトス、エコー・ドライブ 1052

(72)発明者 渡邊 坦
神奈川県川崎市麻生区王禅寺1099番地 株
式会社日立製作所 システム開発研究所内

【公報種別】特許法第17条の2の規定による補正の掲載
【部門区分】第6部門第3区分
【発行日】平成14年7月19日(2002. 7. 19)

【公開番号】特開平7-281897
【公開日】平成7年10月27日(1995. 10. 27)
【年通号数】公開特許公報7-2819
【出願番号】特願平7-108056
【国際特許分類第7版】

G06F 9/42 330
9/46 313

【FI】

G06F 9/42 330 R
9/46 313 B

【手続補正書】

【提出日】平成14年3月19日(2002. 3. 19)

【手続補正1】

【補正対象書類名】明細書

【補正対象項目名】特許請求の範囲

【補正方法】変更

【補正内容】

【特許請求の範囲】

【請求項1】コンパイラの介在なしに、デジタルコンピュータにおいてプロシージャにレジスタを動的に割り当てる方法であって、

複数のスタックレジスタを有する論理レジスタスタックを定義するステップと、

前記論理レジスタスタックを物理レジスタセットにマッピングするためのオフセットを定義するために、ローカルリロケーション項(1rel)を初期化するステップと、

第1のスタックポインタ値(TOL)を初期化することにより、第1のプロシージャによって指定される任意の数のスタックレジスタをローカルレジスタとして該第1のプロシージャに割り当てて、前記論理レジスタスタックにおいて該ローカルレジスタの範囲を区切るステップと、

第1の循環レジスタポインタ値(BOR)および第2の循環レジスタポインタ値(TOR)を、前記第1のスタックポインタ値(TOL)に初期化するステップと、前記第1のプロシージャによって循環レジスタとして指定されたレジスタの任意の数だけ、前記第2の循環レジスタポインタ値(TOR)および前記第1のスタックポインタ値(TOL)をインクリメントすることにより、レジスタを、循環レジスタとして前記第1のプロシージャに割り当てるステップと、

前記第1のプロシージャから復帰するに先立って、前記第2の循環レジスタポインタ値(TOR)および前記第

1のスタックポインタ値(TOL)を、該第1のプロシージャの循環レジスタの数だけデクリメントすることにより、該循環レジスタの割り当てを解除するステップと、

前記第1のプロシージャの実行中のレジスタアクセス操作に関連して、前記ローカルリロケーション項(1rel)にตอบสนองして物理アドレス(r)を求めることにより、ローカルレジスタのそれぞれの論理レジスタアドレス(R)を、前記物理レジスタにマッピングするステップと、を含み、

前記物理レジスタにマッピングするステップは、

(a) 前記論理レジスタアドレス(R)がスタティックレジスタの範囲内にあるならば、該論理レジスタアドレス(R)に等しくなるよう前記物理アドレスをセットし、

(b) 前記論理レジスタアドレス(R)が前記スタティックレジスタの範囲内になく、かつ該論理レジスタが循環レジスタでないならば、前記論理レジスタ(R)および前記ローカルリロケーション項(1rel)を、前記任意の数のスタックレジスタを法とし、前記スタティックレジスタの範囲だけオフセットされたモジュール加算を実行することで、前記物理アドレスをセットし、

(c) 前記論理レジスタアドレス(R)がスタティックレジスタの範囲内になく、かつ前記論理レジスタが循環レジスタならば、前記論理レジスタアドレス(R)が、前記TORおよびBORによって定義される循環レジスタの範囲内にあるかどうかを判断し、

(c-1) 前記論理アドレスRが前記循環レジスタの範囲内にあるならば、前記論理レジスタアドレス(R)および循環リロケーション項(rrel)のモジュール加算したものに前記物理アドレスをセットし、該循環リロケーション項(rrel)は、循環レジスタベース値(RRB)に前記ローカルリロケーション項(1rel)を加算したものに等しく、

(c-2) 前記論理レジスタアドレス (R) が前記循環レジスタの範囲内にないならば、前記論理レジスタアドレス (R) および別の循環リロケーション項 (# r r e l) のモジュール加算したものに前記物理アドレスをセットし、該別の循環リロケーション項 (# r r e l) は、前記循環レジスタベース値 (R R B) から前記循環レジスタの範囲を差し引いたものを前記ローカルリロケーション項 (l r e l) に加算したものに等しい、プロシージャにレジスタを動的に割り当てる方法。

【請求項2】前記第1のスタックポインタ値 (T O L) をストアして、第2のスタックポインタ値 (O T O L) を形成するステップと、

前記第1のスタックポインタ値 (T O L) をインクリメントすることにより、前記第1のプロシージャによって指定される任意の数の追加スタックレジスタを、パラメータ引き渡しレジスタとして該第1のプロシージャに割り当てて、該パラメータ引き渡しレジスタを含むようにするステップと、

呼び出されたプロシージャが参照するために、前記割り当てられたパラメータ引き渡しレジスタに少なくとも1つのパラメータをストアするステップと、をさらに含む、

前記少なくとも1つのパラメータをストアする前記ステップは、前記ローカルリロケーション項に回答して、前記パラメータ引き渡しレジスタを前記物理レジスタセットにマッピングすることを含む、請求項1に記載のプロシージャにレジスタを動的に割り当てる方法。

【請求項3】第2のプロシージャを呼び出すステップと、

前記第1のプロシージャのパラメータ引き渡しレジスタを有する初期ローカルレジスタスペースを前記第2のプロシージャに割り当て、該レジスタにストアされた前記少なくとも1つのパラメータを、メモリ参照なしに前記第2のプロシージャに利用可能なようにするステップと、

前記スタックポインタ値をインクリメントすることにより、前記第2のプロシージャによって指定される任意の数の追加のスタックレジスタを、ローカルレジスタとして該第2のプロシージャに割り当て、前記第1のプロシージャのローカルレジスタの内容をはじめにメモリにセーブすることなしに、該第2のプロシージャのローカルレジスタを含むようにするステップと、

前記第2のプロシージャから復帰する時、ローカルレジスタの数だけ前記スタックポインタ値をデクリメントすることによって、該ローカルレジスタの割り当てを解除し、これによって、ローカルレジスタの内容をセーブおよび復元することなく、該第2のプロシージャを呼び出し、またそこから復帰するステップと、

をさらに含む、請求項2に記載のプロシージャにレジスタを動的に割り当てる方法。

【請求項4】前記第2のプロシージャを呼び出す時、前記第1および第2のスタックポインタ値 (T O L、O T O L) をストアして、該第2のプロシージャから復帰する際に参照するためのストアされた値を形成するステップをさらに含む、

前記割り当てを解除するステップは、前記第1および第2のスタックポインタ値を前記ストアされた値にリセットすることを含む、請求項3に記載のプロシージャにレジスタを動的に割り当てる方法。

【請求項5】追加のパラメータレジスタを前記第1のプロシージャに割り当てるステップをさらに含む、

前記第1および第2のスタックポインタ値をストアする前記ステップは、前記第2のプロシージャからの復帰の際に参照するために、スタックポインタオフセット値を前記追加のパラメータレジスタにストアすることを含む、請求項4に記載のプロシージャにレジスタを動的に割り当てる方法。

【請求項6】前記割り当てを解除するステップは、前記追加のパラメータレジスタにストアされた前記スタックポインタオフセット値からスタックポインタ値を計算し、該計算された値に、前記スタックポインタ値をリセットすることを含む、請求項5に記載のプロシージャにレジスタを動的に割り当てる方法。

【請求項7】前記第2のプロシージャにおいて、前記第1のプロシージャのパラメータ引き渡しレジスタのうちの選択された一つに、計算された値をストアすることによって、該計算された値を前記第1のプロシージャに返すステップをさらに含む、請求項3に記載のプロシージャにレジスタを動的に割り当てる方法。

【請求項8】スタティックレジスタセットを提供し、前記第2のプロシージャにおいて、該スタティックレジスタの選択された一つに、計算された値をストアすることによって、該計算された値を前記第1のプロシージャに返すステップをさらに含む、

これにより、前記パラメータレジスタが、次のプロシージャ呼び出しに即座に利用可能なようにする、請求項3に記載のプロシージャにレジスタを動的に割り当てる方法。

【請求項9】レジスタファイルポートをアクセスする物理アドレスを提供するためのレジスタファイルポートアクセス装置であって、

現プロシージャから仮想アドレス (R) を受け取る入力手段と、

前記仮想アドレス (R) を予め決められた定数と比較して、該仮想アドレスがスタックレジスタのスタティックレジスタを示すかどうかを判断する第1のコンパレータ手段と、

前記仮想アドレスをローカルリロケーション項 (l r e l) に加算して、第1の物理アドレスを形成する手段と、

前記仮想アドレスおよび前記第1の物理アドレス (r) のうちの一つを選択して、該選択されたアドレスを前記レジスタファイルポートに接続する第1のマルチプレクサ手段と、

前記第1のマルチプレクサ手段に結合され、前記仮想アドレスがスタックレジスタを示すならば前記第1の物理アドレスを選択し、前記仮想アドレスがスタティックレジスタを示すならば該仮想アドレスを選択し、これにより、スタックレジスタ参照を、前記現プロシーダに割り当てられた物理レジスタアドレスに振り替える制御手段と、

前記仮想アドレス (R) を第1および第2の循環レジスタポインタ値 (BOR, TOR) と比較し、該仮想アドレスが、前記現プロシーダに循環レジスタとして割り当てられたレジスタを示すかどうかを判断する第2のコンパレータ手段と、

前記仮想アドレス (R) を第1の循環リロケーション項 (rrel) に加算して、第1の物理アドレスを形成する手段と、

前記仮想アドレス (R) を第2の循環リロケーション項 (rrel#) に加算して、第2の物理アドレスを形成

する手段と、

前記第1および第2の物理アドレスのうちの一つを選択し、該選択されたアドレスを、前記レジスタファイルポートアドレス端子に結合する第2のマルチプレクサ手段と、

前記第2のマルチプレクサ手段を制御して、前記仮想アドレス (R) が、前記循環レジスタセット内の回り込みを示さなければ前記第1の物理アドレスを選択し、前記仮想アドレスが前記循環レジスタ内の回り込みを示すならば前記第2の物理アドレスを選択する制御手段と、を備え、

前記第1の循環リロケーション項 (rrel) は、前記ローカルリロケーション項 (lrel) に前記循環レジスタのベース値 (RRB) を加算したものに等しく、前記第2の循環リロケーション項 (rrel#) は、前記循環レジスタベース値 (RRB) から前記循環レジスタセットのサイズを差し引いたものを、前記ローカルリロケーション項 (lrel) に加算したものに等しく、これにより、前記循環レジスタセット内の回り込みを調整する、

レジスタファイルポートアクセス装置。

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

☐ **BLACK BORDERS**

☒ **IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**

☒ **FADED TEXT OR DRAWING**

☐ **BLURRED OR ILLEGIBLE TEXT OR DRAWING**

☐ **SKEWED/SLANTED IMAGES**

☐ **COLOR OR BLACK AND WHITE PHOTOGRAPHS**

☒ **GRAY SCALE DOCUMENTS**

☐ **LINES OR MARKS ON ORIGINAL DOCUMENT**

☐ **REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**

☐ **OTHER:** _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.